Search Forms
Search Results
Help
User Searches
Preferences
Logout

Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term:

Display: 10 Documents in Display Format: - Starting with Number 1

Generate: ○ Hit List ◉ Hit Count ○ Side by Side ○ Image

Search Clear Interrupt

Search History

DATE: Thursday, April 28, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L30	5854933.pn.	1	L30
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L29	L27 and (java or c or c++ or fortran or cobol)	33	L29
L28	L27 and macro	0	L28
L27	L25 and uncompiled near code	33	L27
L26	L25 and (uncompiled or uncompil$) near program with code	15	L26
L25	(pre-processor near macro or preprocessor near macro or runtime or run adj time or run with time or run-time)	161109	L25
L24	(pre-processor near macro or preprocessor near macro)	114	L24
L23	L22 and (extend or expand)	114	L23
L22	l9 and (macro near language or macro with language)	225	L22
L21	715/209	0	L21
L20	707/209	7	L20
L19	717/140	439	L19
L18	717/124	605	L18

L17	717/122	152	L17
L16	717/118	234	L16
L15	717/117	77	L15
L14	717/116	440	L14
L13	717/115	128	L13
L12	717/114	314	L12
L11	717/106	327	L11
L10	717/209	0	L10
L9	717.clas.	8349	L9
L8	715/526	278	L8
L7	715/513	1182	L7
L6	715/500	548	L6
L5	715.clas.	21180	L5
L4	707.clas.	26372	L4
L3	707/100	5967	L3
L2	707/2	4337	L2
L1	707/1	6637	L1

END OF SEARCH HISTORY

D23: Entry 114 of 114 File: USPT Feb 14, 1984

US-PAT-NO: 4432051
DOCUMENT-IDENTIFIER: US 4432051 A

TITLE: Process execution time accounting system

DATE-ISSUED: February 14, 1984

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bogaert; Jean-Louis	Clamart			FR
deRivet; Philippe-Hubert	Paris			FR
Franklin; Benjamin S.	Cambridge	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Honeywell Information Systems Inc.	Waltham	MA			02

APPL-NO: 06/ 202423 [PALM]
DATE FILED: October 30, 1980

PARENT-CASE:
This application is a continuation of U.S. patent application Ser. No. 055,200 filed Jan. 22, 1979 and now abandoned, which was a continuation of U.S. patent application Ser. No. 823,315, filed Aug. 10, 1977 and now abandoned, which was a continuation of U.S. patent application Ser. No. 529,016, filed Dec. 2, 1974, and now abandoned.

FOREIGN-APPL-PRIORITY-DATA:

COUNTRY	APPL-NO	APPL-DATE
FR	73 42702	November 30, 1973

INT-CL: [03] G06F 3/00, G06F 9/22

US-CL-ISSUED: 364/200
US-CL-CURRENT: 717/127; 718/102

FIELD-OF-SEARCH: 364/2MSFile, 364/9MSFile

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	3639912	February 1972	Campbell	364/200
☐	3723975	March 1973	Kurtz, Jr. et al.	364/200
☐	3771144	November 1973	Belady et al.	364/200
☐	3818458	June 1974	Deese	364/200

ART-UNIT: 232

PRIMARY-EXAMINER: Springborn; Harvey E.

ATTY-AGENT-FIRM: Grayson; George Prasinos; Nicholas Gaybrick; Robert J.

ABSTRACT:

A time accounting system for accounting for the time a process spends in a ready state, a wait state, or a running state. The system includes a time-of-day clock coupled to a central processing unit for outputting the time-of-day whenever a process changes state. A memory also coupled to the central processing unit stores the contents of a plurality of addressable process control blocks and each process control block includes first, second, and third storage locations for storing indications of the amount of time an associated process has been in the running, ready, and wait states, respectively. The central processor unit accesses the process control blocks and updates the process times stored therein in accordance with control signals generated by decoding a string of microinstructions stored in a control store memory. The time-of-day clock is accessed each time a process enters one of the running, ready, or wait states and each time the execution of a process is completed. The process control blocks are accessed and updated with the differences in the access times of the time-of-day clock to accumulate in the appropriate storage locations in each process control block the amount of time the associated process was in the running, ready, or wait states.

4 Claims, 57 Drawing figures

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#
Search Forms
Search Results
Help
User Searches
Preferences
Logout
Generate Collection Print

L23: Entry 112 of 114 File: USPT Apr 10, 1990

US-PAT-NO: 4916610
DOCUMENT-IDENTIFIER: US 4916610 A

TITLE: Multilanguage software integration through preprocessing

DATE-ISSUED: April 10, 1990

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bapat; Subodh	Fort Lauderdale	FL		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Racal Data Communications Inc.	Sunrise	FL			02

APPL-NO: 07/ 253464 [PALM]
DATE FILED: October 5, 1988

INT-CL: [04] G06F 9/00

US-CL-ISSUED: 364/300
US-CL-CURRENT: 717/141; 717/114, 717/139, 717/146

FIELD-OF-SEARCH: 364/200, 364/300, 364/900

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4309756	January 1982	Beckler	364/300
☐	4315315	February 1982	Kossiakoff	364/300
☐	4330822	May 1982	Dodson	364/200
☐	4374408	February 1983	Bowles et al.	364/200
☐	4414629	November 1983	Waite	364/300
☐	4463423	July 1984	Potash et al.	364/300
☐	4559614	December 1985	Peek et al.	364/900
☐	4566078	January 1986	Crabtree	364/900
	4567574	January 1986	Saade et al.	364/900

☐				
☐	4595981	June 1986	Leung	364/300
☐	4692896	September 1987	Sakoda et al.	364/900
☐	4719564	January 1988	Hara	364/200

OTHER PUBLICATIONS

Advanced Turbo Pascal Programming & Techniques, Herbert Schildt, 1986, McGraw Hill, pp. 108-124.
CTIX Operating System Manual, Convergent Technologies, date unknown, pp. 1-2.
The C Programming Language, Brian W. Kerninghan and Dennis M. Ritchie, 1978, Prentice-Hall, Inc., pp. 86-87 and 207-208.

ART-UNIT: 232

PRIMARY-EXAMINER: Zache; Raulfe B.

ATTY-AGENT-FIRM: Miller; Jerry A.

ABSTRACT:

A method of assuring consistency of constants in a multilanguage software system, includes generating a first set of code written in a first language using a plurality of symbolic constants to represent a corresponding plurality of actual constants. A second set of code is written in a second language using the plurality of symbolic constants to represent the corresponding plurality of actual constants. A common header file is generated which contains information which relates the plurality of symbolic constants to the corresponding plurality of actual constants. The header is included within the first and second sets of code. The symbolic constants in the first and second sets of code are replaced with their corresponding actual constants during a preprocessing step. Any constructs which are not a part of the first language are stripped from the second set of code including the header file. Any constructs which are not a part of the second language are stripped from the first set of code including the header. The resulting files have their symbolic constants consistently replaced by constants which are defined in the header so that changes need only be made in the header file to assure consistency in the several languages.

11 Claims, 1 Drawing figures

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#
End of Result Set

☐ Generate Collection Print

L29: Entry 33 of 33 File: USPT Dec 28, 1999

US-PAT-NO: 6009273
DOCUMENT-IDENTIFIER: US 6009273 A

TITLE: Method for conversion of a variable argument routine to a fixed argument routine

DATE-ISSUED: December 28, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Ayers; Andrew E.	Amherst	NH		
Liu; Jiyang	Westford	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Hewlett-Packard Company	Palo Alto	CA			02

APPL-NO: 08/ 953549 [PALM]
DATE FILED: October 21, 1997

PARENT-CASE:
This Application claims priority from U.S. Provisional patent application Ser. No. 60/047,866, filed May 29, 1997.

INT-CL: [06] G06 F 9/44

US-CL-ISSUED: 395/709; 395/708, 395/704, 395/705, 712/226, 712/234, 712/241
US-CL-CURRENT: 717/143; 712/226, 712/234, 712/241, 717/157, 717/160

FIELD-OF-SEARCH: 295/709, 295/708, 295/704, 295/705, 295/567, 295/580, 295/581, 295/588, 712/226, 712/233, 712/234, 712/241

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5740443	April 1998	Carini	395/705
☐	5854933	December 1998	Chang	395/709

OTHER PUBLICATIONS

Kaser et al., "On the Conversion of Indirect to Direct Recursion," ACM Letters on Programming Languages and Systems, vol. 2, No. 1-4, pp. 151-164, Mar. 1993.
Frank Vahid, "Procedure Cloning: A Transformation for Improved System-Level Functional Partitioning," Proceedings of European Design and Test Conference, ED&TC 97, pp. 487-492, Mar. 1997.
Cooper et al., "Procedure Cloning," Proceedings of the 1992 International Conference on Computer Languages, pp. 96-105, Apr. 1992.
Cooper et al., "Unexpected Side Effects of Inline Substitution: A Case Study," ACM Letters on Programming Languages and Systems, vol. 1, No. 1, pp. 22-32, Mar. 1992.

ART-UNIT: 272

PRIMARY-EXAMINER: Hafiz; Tariq R.

ASSISTANT-EXAMINER: Dam; Tuan Q.

ABSTRACT:

A compiler method analyzes a program listing to identify a first set of subroutines therein, each of which accepts a variable number of arguments, converting the first set of subroutines into further sets of subroutines which accept fixed numbers of arguments. The method includes the steps of: locating subroutines in the program listing which accept a variable number of arguments and identifying which thereof comprise a first set of subroutines that can be altered to a form which accepts a fixed number of arguments; for each subroutine identified as part of the first set, determining call sites which pass arguments to each subroutine and determining a number and kind of said arguments to be passed therefrom; partitioning call sites to each subroutine of the first set into one or more groups, each group comprising call sites which pass an identical number and kind of arguments to an associated subroutine of said first set; duplicating each subroutine of said first set into plural second sets of N corresponding subroutines, wherein N is equal to the number of groups associated with the subroutine of the first set, and revising each one of said N corresponding subroutines to receive a number of fixed arguments to be passed by call sites partitioned into an associated group; and substituting in the program listing the N corresponding subroutines and revising call sites in each group to refer to an associated one of said N corresponding subroutines.

8 Claims, 5 Drawing figures

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#
Search Forms
Search Results
Help
User Searches
Preferences
Logout
Generate Collection Print

L24: Entry 97 of 114 File: USPT Oct 21, 1997

US-PAT-NO: 5680585
DOCUMENT-IDENTIFIER: US 5680585 A

TITLE: Method and apparatus for defining data packet formats

DATE-ISSUED: October 21, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bruell; Gregory O.	Carlisle	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Bay Networks, Inc.	Santa Clara	CA			02

APPL-NO: 08/ 414445 [PALM]
DATE FILED: March 31, 1995

INT-CL: [06] G06 F 13/00, G06 F 13/42, G06 F 15/00

US-CL-ISSUED: 395/500; 364/240.8, 364/260.1, 364/280.4, 364/284
US-CL-CURRENT: 703/26

FIELD-OF-SEARCH: 395/200.18, 395/200.2, 395/500, 395/375, 395/800

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5021947	June 1991	Campbell et al.	395/800
☐	5475838	December 1995	Fehskens	395/185.1
☐	5483640	January 1996	Isfeld et al.	395/200.03
☐	5515508	May 1996	Pettus et al.	395/200.01
☐	5524253	June 1996	Pham et al.	395/800

ART-UNIT: 234

PRIMARY-EXAMINER: Teska; Kevin J.

ASSISTANT-EXAMINER: Mohamed; Ayni

ATTY-AGENT-FIRM: Blakely, Sokoloff, Taylor & Zafman

ABSTRACT:

A packet description language which is declarative in nature and suitable for efficiently and flexibly defining data packet formats in accordance with internetwork routing device uses. Data packet formats may be defined utilizing the packet description language and then compiled to create a data structure corresponding to the defined data packet format. A routing device test platform may generate test data packets and decode received test packets by referencing the test data to the compiled data structure defined in accordance with the packet description language. The declarative language provides for assigning numerous default values and attributes to packet fields such that only a small amount of data need be specified when regression testing a new routing device.

14 Claims, 5 Drawing figures

Previous Doc Next Doc Go to Doc#

L24: Entry 102 of 114 File: USPT Jan 4, 1994

US-PAT-NO: 5276880
DOCUMENT-IDENTIFIER: US 5276880 A

TITLE: Method for parsing and representing multi-versioned computer programs, for simultaneous and synchronous processing of the plural parses

DATE-ISSUED: January 4, 1994

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Platoff; Michael A.	Monmouth Junction	NJ		
Wagner; Michael E.	Hopewell Township, Mercer County	NJ		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Siemens Corporate Research, Inc.	Princeton	NJ			02

APPL-NO: 07/ 451493 [PALM]
DATE FILED: December 15, 1989

INT-CL: [05] G06F 9/45

US-CL-ISSUED: 395/700; 364/280, 364/280.4, 364/280.5, 364/DIG.1
US-CL-CURRENT: 717/143; 717/154

FIELD-OF-SEARCH: 364/DIG.1, 364/DIG.2, 395/700

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5050068	September 1991	Dollas et al.	395/375
☐	5105353	April 1992	Charles et al.	395/700
☐	5151991	September 1992	Iwasawa et al.	395/700

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0343883	November 1989	EP	

OTHER PUBLICATIONS

"The Theory and Practice of Compiler Writing", Tremblay et al., McGraw-Hill Book Company, pp. 1-13.
"Using Tomorrow's C Standard Today", Williams et al., Computer Language, Jul. 1988, vol. 5, pp. 49-54.
"A Coroutine Approach to Parsing", Hanan Samet, ACM Transactions on Programming Languages and Systems, vol. 2, No. 3, Jul. 1980, pp. 290-306.
Patent Abstracts of Japan, vol. 9, No. 214 (P-384) Aug. 31, 1985 & JP-A-60 074 039 (Fujitsu K. K.) Apr. 26, 1985.
"Bilingual Mapping Macro", T. L. Oliver, IBM Technical Disclosure Bulletin, vol. 13, No. 10, Mar. 1971, p. 2903.

ART-UNIT: 237

PRIMARY-EXAMINER: Lee; Thomas C.

ASSISTANT-EXAMINER: Von Buhr; Maria N.

ATTY-AGENT-FIRM: Ahmed; Adel A.

ABSTRACT:

A modified parser generator enables the parsing of programs such as C programs with preprocessor directives and implemented extensions to the normal abstract syntax tree representation of programs to create an integrated representation structure.

21 Claims, 3 Drawing figures

Previous Doc Next Doc Go to Doc#

L24: Entry 104 of 114 File: USPT Jun 5, 1990

US-PAT-NO: 4931928
DOCUMENT-IDENTIFIER: US 4931928 A
** See image for Certificate of Correction **

TITLE: Apparatus for analyzing source code

DATE-ISSUED: June 5, 1990

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Greenfeld; Norton R.	Wayland	MA	01778	

APPL-NO: 07/ 269135 [PALM]
DATE FILED: November 9, 1988

INT-CL: [05] G06F 9/44

US-CL-ISSUED: 364/300
US-CL-CURRENT: 717/131; 707/3, 717/114, 717/142, 717/143

FIELD-OF-SEARCH: 364/200, 364/300, 364/900

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4506326	March 1985	Shaw et al.	364/300
☐	4686623	August 1987	Wallace	364/300
☐	4688195	August 1987	Thompson et al.	364/300
☐	4729096	March 1988	Larson	364/300
☐	4751635	June 1988	Kret	364/200
☐	4787035	November 1988	Bourne	364/300

OTHER PUBLICATIONS

"Global Program Analysis In An Interactive Environment" by Larry M. Masinter, SSL-80-1 Xerox Palo Alto Research Center, Palo Alto, Calif., Jan. 1980, pp. 39-61 and 67-83 (Chapter 4-6 and Apendixes 1-3 respectively).
"Telescope: A Cross Reference Utility for Lisp" by Jed Krohnfeldt, Utah PASS Project Op Note 86-11, Dec. 4, 1986, pp. 1-2.

"Reverserver: Databases for Reverse Engineering" in Release 1.0, published by EDventure Holding, Inc., Apr. 10, 1989, pp. 12-13.

ART-UNIT: 232

PRIMARY-EXAMINER: Zache; Raulfe B.

ATTY-AGENT-FIRM: Hamilton, Brook, Smith & Reynolds

ABSTRACT:

Apparatus in a computer system provides source code analysis. The apparatus includes an analysis member which extracts programming semantics information from an input source code. The analysis member operates according to the programming language of the source code as defined by a grammar mechanism. The analysis member employs a database interface which enables the extracted programming semantics information to be placed in a user desired database for subsequent recall by a desired query system. The database and query system may be pre-existing elements which are supported by a digital processor independently of the analysis member. A relational database with an SQL query system may be used.

19 Claims, 8 Drawing figures

Previous Doc Next Doc Go to Doc#

L24: Entry 105 of 114 File: JPAB Apr 10, 1998

PUB-NO: JP410091166A
DOCUMENT-IDENTIFIER: JP 10091166 A
TITLE: ACOUSTIC MODEL GENERATION DEVICE AND PROGRAM OPTIMIZING METHOD IN ACOUSTIC MODEL GENERATION DEVICE

PUBN-DATE: April 10, 1998

INVENTOR-INFORMATION:
NAME COUNTRY
CHARLES, BRIGHT
FUJIWARA, SHUNGO
SOMA, DAIZABURO

ASSIGNEE-INFORMATION:
NAME COUNTRY
KK KORUGU

APPL-NO: JP08266624
APPL-DATE: September 18, 1996

INT-CL (IPC): G10 H 5/04; G10 H 1/02

ABSTRACT:

PROBLEM TO BE SOLVED: To enable increasing functions of a program by retrenching and reducing needless and redundant programs by optimization processing of a program.

SOLUTION: A processor depending section is constituted with a macro- preprocessor 21 converting a macro-preprocessor input list 20 to a form being able to compile, a compiler 22 translating a processed result of the macro- preprocessor 21 to an acoustic synthesis program being performable, and an optimizer 23 performing optimization processing of a program. Indication of execution is performed in order for the macro-preprocessor 21, the compiler 22, and the optimizer 23 through a lock edition tool 18. And the macro- preprocessor input list 20 is converted to a form being able to compile, and the processed result is compiled in the acoustic synthesis program being performable. Further, processing for reducing needless steps and redundant steps out of the acoustic synthesis program being performable is performed.

TDBD-ACC-NO: NN9503641

DISCLOSURE TITLE: Automated Support Code Generation for an Object Oriented System

PUBLICATION-DATA:
IBM Technical Disclosure Bulletin, March 1995, US

VOLUME NUMBER: 38
ISSUE NUMBER: 3
PAGE NUMBER: 641 - 646

PUBLICATION-DATE: March 1, 1995 (19950301)

CROSS REFERENCE: 0018-8689-38-3-641

DISCLOSURE TEXT:

A method of generating supporting source code based on information extracted from a project's Object Oriented Analysis (OOA) models is disclosed. Code generation can be automated and controlled by an underlying architecture domain to help achieve the desired system performance and flexibility goals while preserving a framework derived from extensive modeling analysis. Excerpts of a functional implementation are used as examples to illustrate the interrelation between application objects and the architecture domain. In addition to basic object characteristics such as object name and object type, other artificially asserted characteristics can be enforced during analysis to improve the modeling representation and design documentation. This facilitates error checking within models and enables automated code generation. The following asserted object characteristics are used for automated code generation: o unique object alias name o requirement for instance list support o requirement for subtype migration support o resident processor o transition control statements o event naming conventions that uniquely define each event o event tracing class Two different types of processing are involved in automated code generation. System wide definitions for data structures, global types, constants, and macros are generated using a model extraction tool set. Object specific definitions and code are generated through subsequent macro preprocessing using a compiler tool set. Given a software tool capable of extracting information from analysis models, a set of local macro preprocessor definitions are generated for each object in a file with a name derived from the object's alias name. These local macro preprocessor definitions define the basic object characteristics and are used to control the generation of specific supporting code during object module compilation. These local macros define the following information in macro language syntax: o object full name o object alias name o object type (Active, Passive, or Monitor) o object storage class and size o object requirement for instance list support o object inheritance level (Supertype, Subtype, or Stand-Alone) o if an object has a supertype, the supertype full name o if an object has a supertype, the supertype alias name Prior to object module compilation, a macro preprocessor is invoked to process the source code files and replace all defined local macros with assigned text strings. This creates specific symbols and code that are only meaningful to a particular object in specific implementation language syntax. As an example, to support a doubly linked instance list for a supertype object, two additional attributes are generated and inserted

into an instance data structure to maintain links between the supertype's instances. To guarantee that these attributes are uniquely defined within an instance data structure, each attribute is prefixed with the specific supertype alias name. The process of generating these attributes uses a single code template for all inserted attributes. An example follows: Prerequisites: local macro: zz_level (object inheritance level) local macro: zz_inst_list (requirement for instance list support) local macro: zz_alias (object alias name) Process of Code Generation for Insertion of Link Attributes: Description: If this object is a supertype object and it requires instance list support, insert two link attributes prefixed with this object's alias name to its instance data structure definition Pseudo Code Template:

Previous Doc Next Doc Go to Doc#

L24: Entry 109 of 114 File: TDBD Jan 1, 1979

TDB-ACC-NO: NN79013356

DISCLOSURE TITLE: General Purpose Assembler For Microprocessors. January 1979.

PUBLICATION-DATA:
IBM Technical Disclosure Bulletin, January 1979, US

VOLUME NUMBER: 21
ISSUE NUMBER: 8
PAGE NUMBER: 3356 - 3357

PUBLICATION-DATE: January 1, 1979 (19790101)

CROSS REFERENCE: 0018-8689-21-8-3356

DISCLOSURE TEXT:

2p. Using macros with existing assemblers for microprocessors presents problems in that either a new assembler has to be written for each new microprocessor or the use of any existing assembler is restricted by the architecture of the original machine for which it was designed. This article describes a General Purpose Assembler (GPA) designed for use with a macro preprocessor and which gives flexible bit handling and error message generation. The assembler is capable of general application and is not restricted to byte orientated architecture. It can be used for microprocessors, micro programming and programmable logic arrays (PLAs). - In order to generate an assembler for a given source language using the GPA. the user must write a series of macros to translate the source language into the powerful low level language understood by the GPA. - The principle of using macros to translate one assembly language into another is not new, but hitherto this approach has always led to problems due to differences in the architectures of the new and old languages. A particular major problem is dealing with error conditions unique to each processor, for example, testing whether branches are in range, etc. - The GPA solves this problem by permitting conditional error messages which can test assemble time values (equated symbols, address counter etc). - The figure shows the general structure of the GPA. The source program may include macros processed by the same processor. The language in which the macros are written depends upon the macro processor, most stand-alone macro processors are suitable (ML/1, GMP could be used). - The temporary file holds the original source records, interspersed with lower level GPA statements. (These statements do not appear on the listing.) The GPA is a single program written in PL/1 and is readily transportable. The GPA assembles the bit strings required and carries out the necessary housework to produce a useful listing. - The listing is a list of source code with error messages, cross references and error totals.

Previous Doc Next Doc Go to Doc#